Fast Rendering of Complex Environments Using a Spatial Hierarchy

Bradford Chamberlain
Tony DeRose
Dani Lischinski
David Salesin
John Snyder*

University of Washington / *Microsoft Research

Graphics Interface '96 -- May 23, 1996

The Problem

Problem: Quickly render complex static scenes

Challenge: Rendering is linear in number of triangles

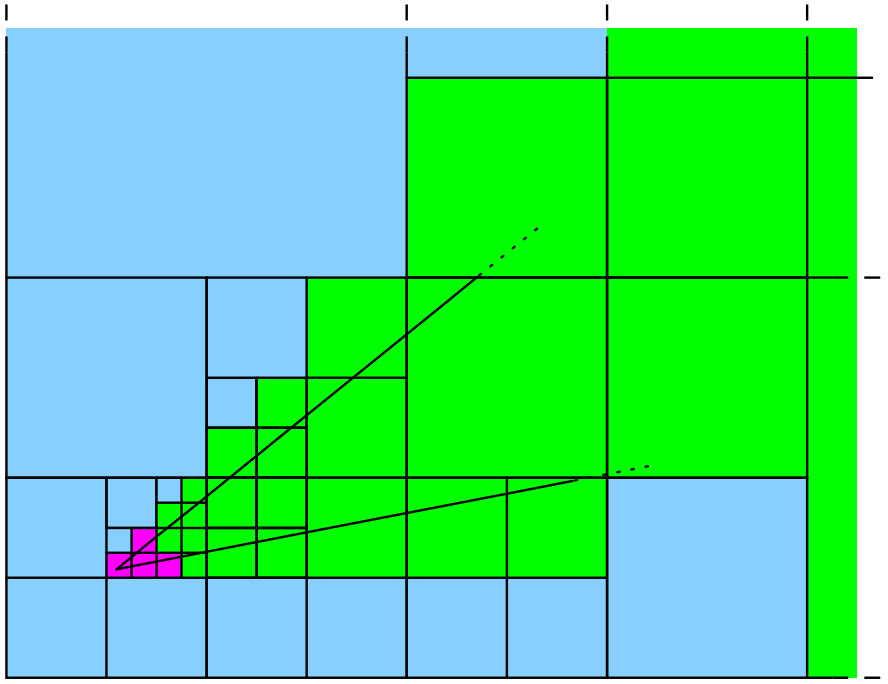
Our Goal: Develop a sublinear rendering algorithm

Related Work

- **Object-based Approaches:**
  - level-of-detail models *(e.g. Funkhouser & Sequin)*
  - object hierarchies *(e.g. Maciel & Shirley)*
- **Spatially-based Approaches:**
  - volume rendering *(e.g. Meagher)*
  - hierarchical splatting *(e.g. Laur & Hanrahan)*

Our Approach

- Create a static octree for the scene
- Create a *color-cube* for each octree cell
- Use color-cubes to approximate distant geometry



Outline

- Implementation
  - Octree Creation
  - Color-cube Creation
  - Rendering the Octree

  ⟩ *Preprocessing*

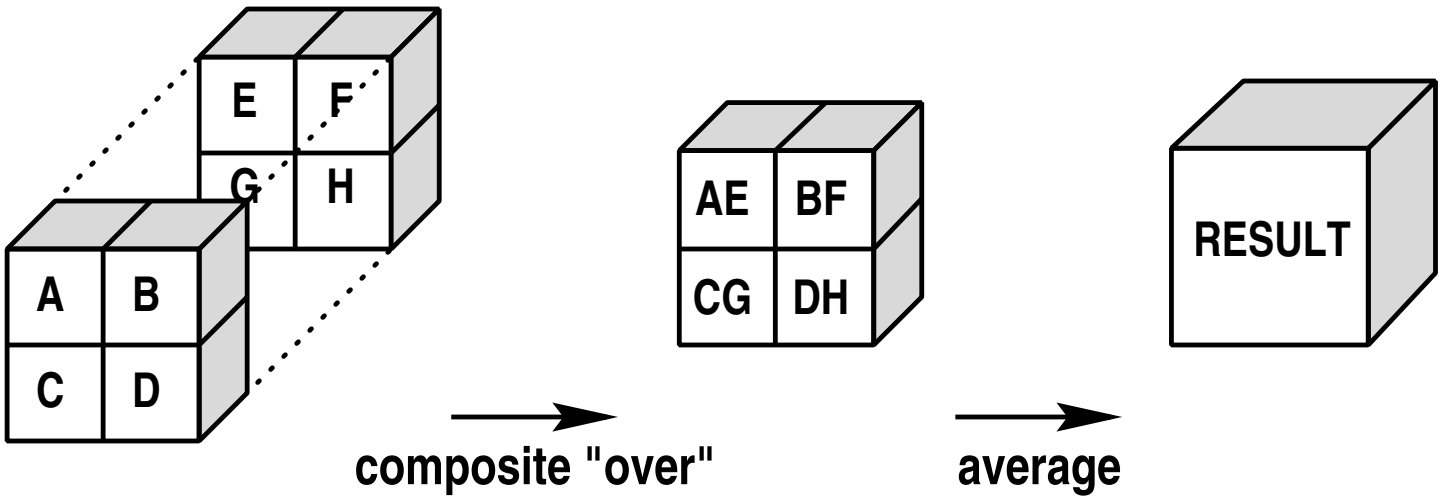
  − *Runtime*
- Sublinearity of Rendering
- Experimental Results
- Video
- Conclusions

Preprocessing: Octree Creation

- Rendering cost assigned to geometric primitives
- All primitives placed in root cell
- If overfull, subdivide and push geometry into children
- Repeat recursively for child cells

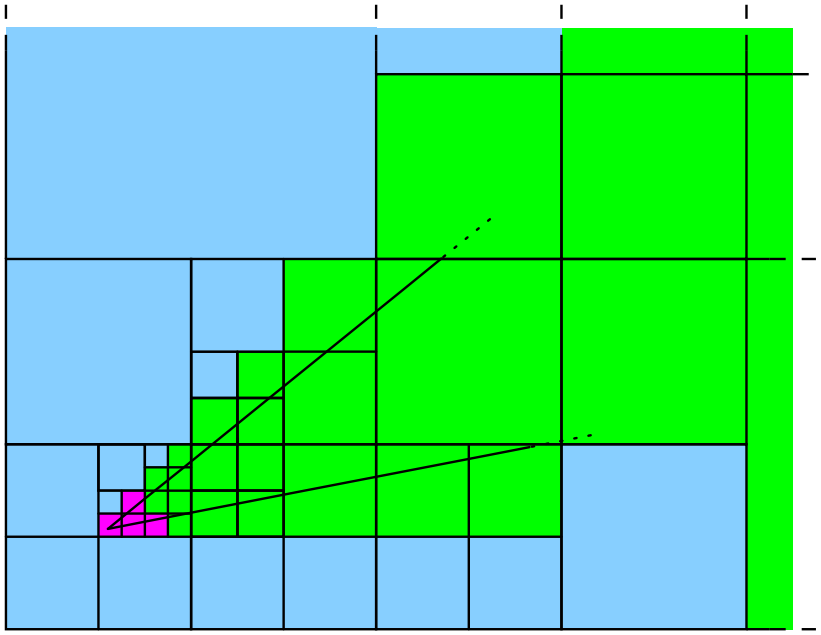
Preprocessing: Color-cube Creation

- Using depth-first octree traversal:
  - Render leaf cell geometry to compute color, opacity
  - Composite and average sibling cubes



Runtime: Rendering the Octree

- Starting with the root cell:
  - if outside the view frustum, ignore
  - if projected size is small, draw color-cube
  - otherwise, consider children recursively
  - if no children, draw contained geometry



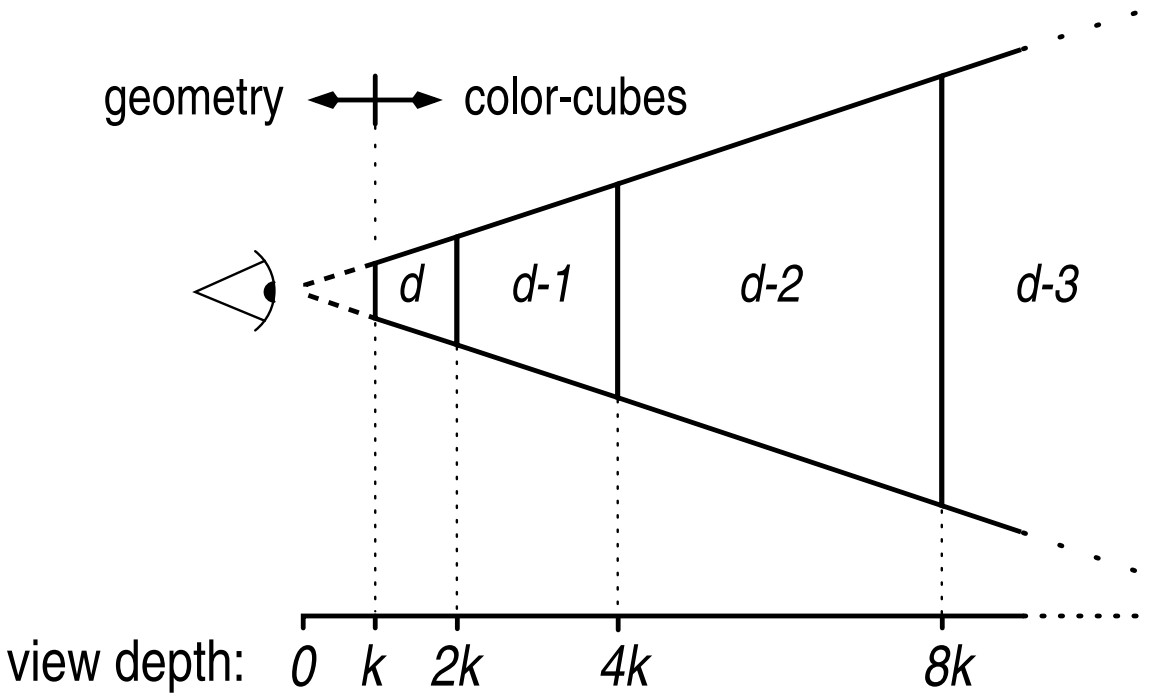
Sublinearity Assumptions

1) Octree depth, d, is O(log N)

2) Triangles per leaf cell is bounded by a constant

Rendering Observation

- View frustum can be thought of as being segmented
- Each sub-frustum corresponds to an octree level

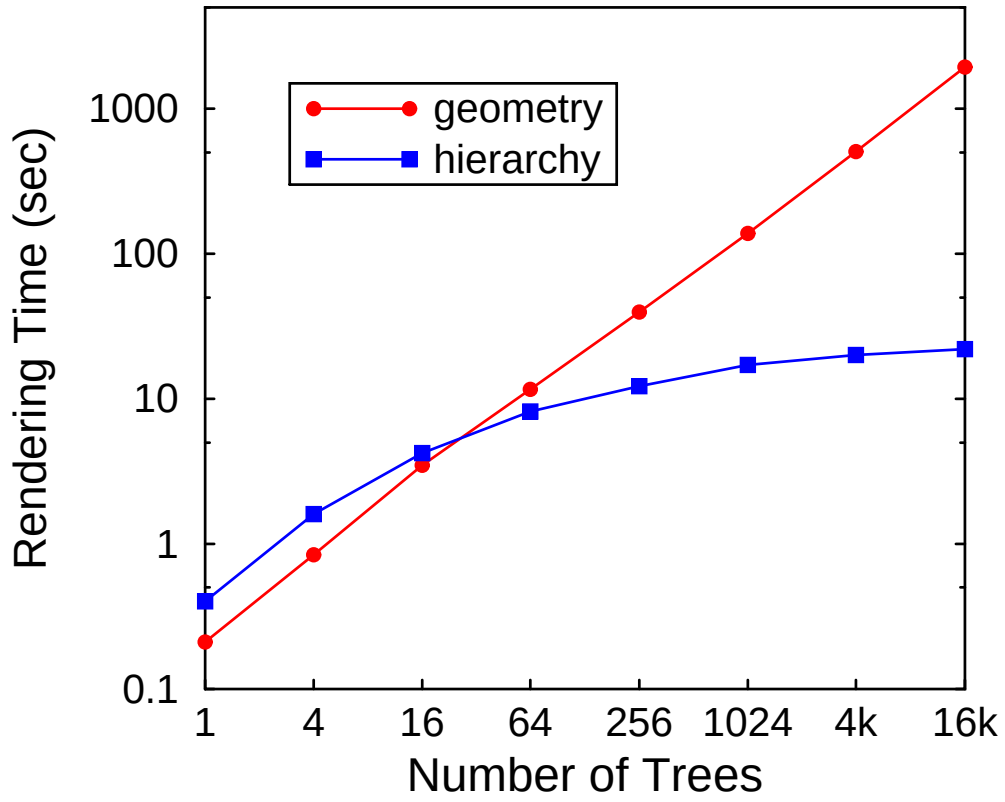


Sublinearity Argument

- Constant number of cells visited at each level
- Each cell is processed in constant time
- Octree depth is assumed to be O(log N)
- Therefore, rendering is O(log N)

Sublinearity Results

- Repeatedly quadruple tree database
- Compare geometric vs. hierarchical times



Drawbacks

- Limitations of only six sampled directions
- Transparency artifacts

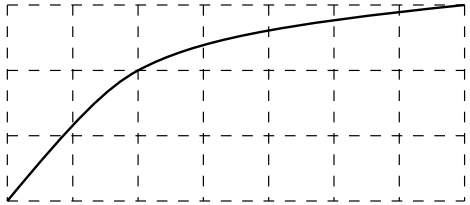
Future Work

- Better octree creation heuristics
- Alternative cell representations
- OS considerations
- Parallel implementation
- Integration with other approaches

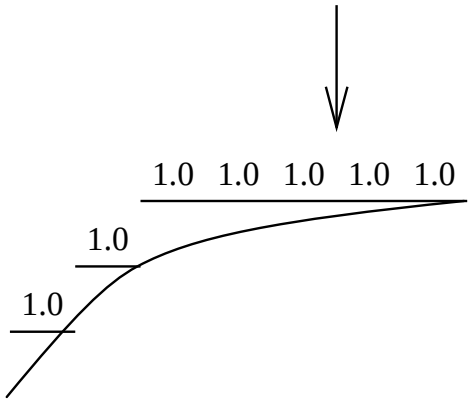
Conclusions

- Our algorithm:
  - uses unstructured databases
  - is simple and extensible
  - works with arbitrary geometric primitives
  - exhibits sublinear rendering times
  - reduces aliasing artifacts

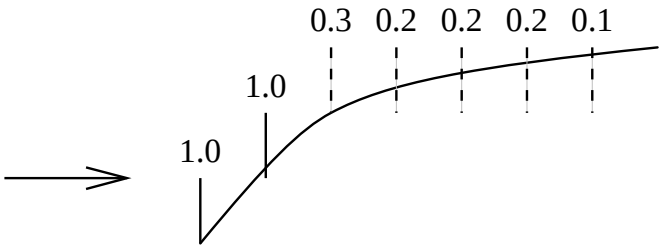
Transparency Artifact



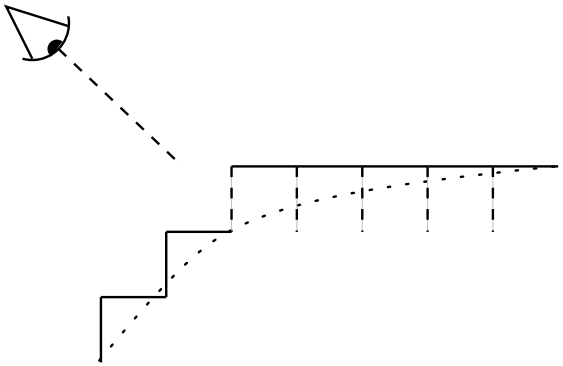
Input Geometry



Sample



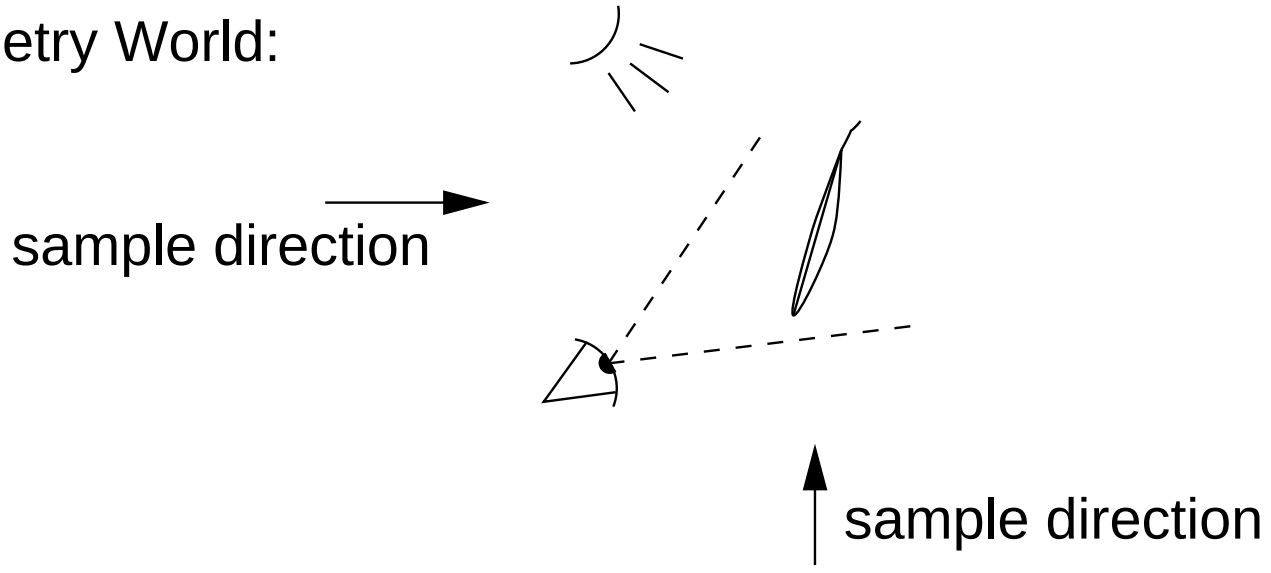
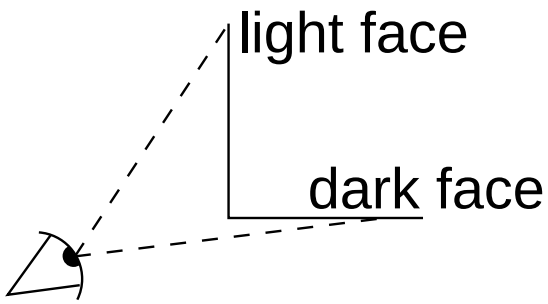
Sample



Off-Axis View

Hidden Surface Artifact

Geometry World:



Hierarchical World: